I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis

Brad Miller[1], Ling Huang[2], A. D. Joseph[1], and J. D. Tygar[1]

[1] UC Berkeley
[2] Intel Labs

Abstract. Revelations of large scale electronic surveillance and data mining by governments and corporations have fueled increased adoption of HTTPS. We present a traffic analysis attack against over 6000 webpages spanning the HTTPS deployments of 10 widely used, industry-leading websites in areas such as healthcare, finance, legal services and streaming video. Our attack identifies individual pages in the same website with 90% accuracy, exposing personal details including medical conditions, financial and legal affairs and sexual orientation. We examine evaluation methodology and reveal accuracy variations as large as 17% caused by assumptions affecting caching and cookies. We present a novel defense reducing attack accuracy to 25% with a 9% traffic increase, and demonstrate significantly increased effectiveness of prior defenses in our evaluation context, inclusive of enabled caching, user-specific cookies and pages within the same website.

1 Introduction

HTTPS is far more vulnerable to traffic analysis than has been previously discussed by researchers. In a series of important papers, a variety of researchers have shown a number of traffic analysis attacks on SSL proxies [1,2], SSH tunnels [3,4,5,6,7], Tor [3,4,8,9], and in unpublished work, HTTPS [10,11]. Together, these results suggest that HTTPS may be vulnerable to traffic analysis. This paper confirms the vulnerability of HTTPS, but more importantly, gives new and much sharper attacks on HTTPS, presenting algorithms that decrease errors 3.9x from the best previous techniques. We show the following novel results:

- Novel attack technique capable of achieving 90% accuracy over 500 pages hosted at the same website, as compared to 60% with previous techniques
- Impact of caching and cookies on traffic characteristics and attack performance, affecting accuracy as much as 17%
- Novel defense reducing accuracy to 25% with 9% traffic increase; significantly increased effectiveness of packet level defenses in the HTTPS context

We evaluate attack, defense and measurement techniques on websites for healthcare (Mayo Clinic, Planned Parenthood, Kaiser Permanente), finance (Wells Fargo, Bank of America, Vanguard), legal services (ACLU, Legal Zoom) and streaming video (Netflix, YouTube).

We design our attack to distinguish minor variations in HTTPS traffic from significant variations which indicate distinct webpages. Minor traffic variations may be caused by caching, dynamically generated content, or user-specific content including cookies. To distinguish minor variations, our attack employs clustering and Gaussian similarity techniques to transform variable length traffic into a fixed width representation. Due to similarity with the Bag-of-Words approach to text analysis, we refer to our technique as Bag-of-Gaussians (BoG). We augment our technique with a hidden Markov model (HMM) leveraging the link structure of the website and further increasing accuracy. Our approach achieves substantially greater accuracy than attacks developed by Panchenko *et al.* (Pan) [8], Liberatore and Levine (LL) [6], and Wang *et al.* [9].[3]

We also present a novel defense technique and evaluate several previously proposed defenses. In the interest of deployability, all defenses we evaluate have been selected or designed to require minimal state. Our evaluation demonstrates that some techniques which are ineffective in other traffic analysis contexts have significantly increased impact in the HTTPS context. For example, although Dyer *et al.* report exponential padding as decreasing accuracy of the Panchenko classifier from 97.2% to 96.6% on SSH tunnels with website homepages [5], we observe a decrease from 60% to 22% in the HTTPS context. Our novel defense reduces the accuracy of the BoG attack from 90% to 25% while generating only 9% traffic overhead.

We conduct our evaluations using a dataset of 463,125 page loads collected from 10 websites during December 2013 and January 2014. Our collection infrastructure includes virtual machines (VMs) which operate in four separate collection modes, varying properties such as caching and cookie retention across the collection modes. By training a model using data from a specific collection mode and evaluating the model using a different collection mode, we are able to isolate the impact of factors such as caching and user-specific cookies on analysis results. We present these results along with insights into the fundamental properties of the traffic itself.

Our evaluation spans four website categories where the specific pages accessed by a user reveal private information. The increased importance of contents over existence of communication is present in traditional privacy concepts such as patient confidentiality or attorney-client privilege. We examine three websites related to healthcare, since the page views of these websites have the potential to reveal whether a pending procedure is an appendectomy or an abortion, or whether a chronic medication is for diabetes or HIV/AIDS. We also examine legal websites, offering services spanning divorce, bankruptcy and wills and legal information regarding LGBT rights, human reproduction and immigration. As documented by Chen *et al.*, specific pages accessed within financial websites may reveal income levels, investment and family details; hence we examine three financial websites [12]. Lastly, we examine two streaming video sites, as the Netflix privacy breach demonstrates the importance of streaming video privacy.

[3] To facilitate further research, code and data from this work are available for download at `http://secml.cs.berkeley.edu/pets2014`.

	Privacy	Page Set	Page Set	Accuracy			Traffic	Analysis	Active
Author	Technology	Scope	Size	(%)	Cache	Cookies	Composition	Primitive	Content
Miller	**HTTPS**	**Closed**	**6388**	**90**	**On**	**Individual**	**Single Site**	**Packet**	**On**
Hintz [1]	SSL proxy	Closed	5	100	?	Individual	Homepages	Request	?
Sun [2]	SSL proxy	Open	2,000 100,000	75 (TP) 1.5 (FP)	Off	Universal	Single Site	Request	Off
Cheng [10]	HTTPS	Closed	489	96	Off	Individual	Single Site	Request	Off
Danezis [11]	HTTPS	Closed	?	89	n/a	n/a	Single Site	Request	n/a
Herrmann [3]	SSH tunnel	Closed	775	97	Off	Universal	Homepages	Packet	?
Cai [4]	SSH tunnel	Closed	100	92	Off	Universal	Homepages	Packet	Scripts
Dyer [5]	SSH tunnel	Closed	775	91	Off	Universal	Homepages	Packet	?
Liberatore [6]	SSH tunnel	Closed	1000	75	Off	Universal	Homepages	Packet	Flash
Bissias [7]	SSH tunnel	Closed	100	23	?	Universal	Homepages	Packet	?
Wang [9]	Tor	Open	100 1000	95 (TP) .06 (FP)	Off	Universal	Homepages	Packet	Off
Wang [9]	Tor	Closed	100	91	Off	Universal	Homepages	Packet	Off
Cai [4]	Tor	Closed	100	78	On	Universal	Homepages	Packet	Scripts
Cai [4]	Tor	Closed	800	70	Off	Universal	Homepages	Packet	Scripts
Panchenko [8]	Tor	Closed	775	55	Off	Universal	Homepages	Packet	Off
Panchenko [8]	Tor	Open	5 1,000	56-73 (TP) .05-.89 (FP)	Off	Universal	Homepages	Packet	Off
Herrmann [3]	Tor	Closed	775	3	Off	Universal	Homepages	Packet	?
Coull [13]	Anonymous Trace	Open	50 100	49 .18	On	Universal	Homepages	NetFlow	Flash & Scripts

Table 1: Prior works have focused almost exclusively on website homepages accessed via proxy. Cheng and Danezis work is preliminary and unpublished. Note that "?" indicates the author did not specify the property.

2 Prior Work

In this section we review attacks and defenses proposed in prior work, as well as the contexts in which work is evaluated. Comparisons with prior work are limited since much work has targeted specialized technologies such as Tor.

Table 1 presents an overview of prior attacks. The columns are as follows:

Privacy Technology The protection mechanism analyzed for traffic analysis vulnerability. Note that some authors considered multiple mechanisms.

Page Set Scope *Closed* indicates the evaluation used a fixed set of pages known to the attacker in advance. *Open* indicates the evaluation used traffic from pages both of interest and unknown to the attacker. Whereas open conditions are appropriate for Tor, closed conditions are appropriate for HTTPS.

Page Set Size For closed scope, the number of pages used in the evaluation. For open scope, the number of pages of interest to the attacker and the number of background traffic pages, respectively.

Accuracy For closed scope, the percent of pages correctly identified. For open scope, the true positive (TP) rate of correctly identifying a page as being within the censored set and false positive (FP) rate of identifying an uncensored page as censored.

Cache *Off* indicates caching disabled. *On* indicates default caching behavior.

Cookies *Universal* indicates that training and evaluation data were collected on the same machine or machines, and consequently with the same cookie values. *Individual* indicates training and evaluation data were collected on separate machines with distinct cookie values.

Traffic Composition *Single Site* indicates the work identified pages within a website or websites. *Homepages* indicates all pages used in the evaluation were the homepages of different websites.

Analysis Primitive The basic unit on which traffic analysis was conducted. *Request* indicates the analysis operated on the size of each object (e.g. image, style sheet, etc.). *Packet* indicates meta-data observed from TCP packets. *NetFlow* indicates network traces anonymized using NetFlow.

Active Content Indicates whether Flash, JavaScript, Java or any other plugins were enabled in the browser.

Several works require discussion in addition to Table 1. Chen *et al.* study side-channel leaks caused by AJAX in web applications. As Chen focuses on traffic generated after a page loads, we view the work as both complimentary and orthogonal [12]. Danezis focused on the HTTPS context, but evaluated his technique using HTTP server logs at request granularity, removing any effects of fragmentation, concurrent connections or pipelined requests [11]. Cheng *et al.* also focused on HTTPS and conducted an evaluation which parsed request sizes from unencrypted HTTP traffic at a website intentionally selected for its static content [10]. Like Cheng and Danezis, Sun *et al.* and Hintz *et al.* assume the ability to parse entire object sizes from traffic [1,2]. For these reasons, we compare our work to Liberatore and Levine, Panchenko *et al.* and Wang *et al.* as these are more advanced and recently proposed techniques.

Herrmann [3] and Cai [4] both conduct small scale preliminary evaluations which involve enabling the browser cache. These evaluations only consider website homepages and all pages are loaded in a fixed, round-robin order. Herrmann additionally increases the cache size from the default to 2GB. We evaluate the impact of caching on pages within the same website, where caching will have a greater effect due to increased page similarity, and load pages in a randomized order for greater cache state variation.

Separate from attacks, we also review prior work relating to traffic analysis defense. Dyer *et al.* conduct a review of low level defenses operating on individual packets [5]. Dyer evaluates defenses using data released by Liberatore and Levine and Herrmann *et al.* which collect traffic from website homepages on a single machine with caching disabled. In this context, Dyer finds that low level defenses are ineffective against attacks which examine features aggregated over multiple packets. Our evaluation finds that low level defenses are considerably more effective in the HTTPS context.

In addition to the packet level defenses evaluated by Dyer, many defenses have been proposed which operate at higher levels with additional cost and implementation requirements. These include HTTPOS [14], traffic morphing [15] and BuFLO [4,5]. HTTPOS manipulates features of TCP and HTTP to affect packet size, object size, pipelining behavior, packet timing and other properties. BuFLO sends a constant stream of traffic at a fixed packet size for a pre-set minimum amount of time. Given the effectiveness and advantages of packet level level defenses in our evaluation context, we do not further explore these higher level approaches in our work.

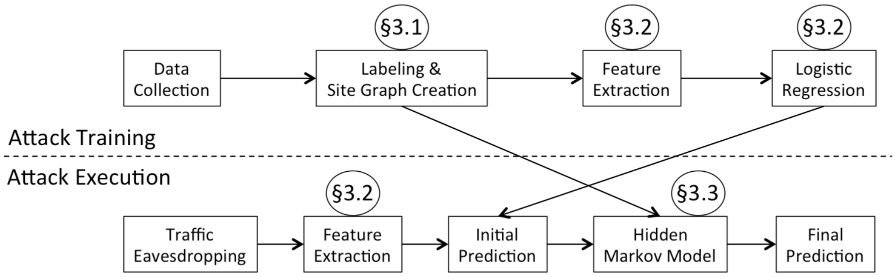


Fig. 1: The dashed line separates training workflow from attack workflow. Bubbles indicate the section in which the system component is discussed. Note that the attacker may conduct training in response to a victim visiting a website, recording victim traffic and inferring contents after browsing has occurred.

3 Attack Presentation

Figure 1 presents the workflow of the attacker as well as the subsections in which we discuss his efforts. In section 3.1, we present a formalism for labeling webpages and generating a link graph relating labeled webpages. Section 3.2 presents the core of our classification approach: Gaussian clustering techniques that capture significant variations in traffic and allow logistic regression to robustly identify objects that reliably differentiate pages. Having generated isolated predictions, we then leverage the site graph and sequential nature of the data in section 3.3 with a hidden Markov model (HMM) to further improve accuracy.

Throughout this section we depend on several terms defined as follows:

Webpage A set of resources loaded by a browser in response to a user clicking a link or entering a URL into the browser address bar. Webpages representing distinct resources are considered *the same* if a user would likely view their contents as substantially similar, regardless of the specific URLs fetched while loading the webpages or dynamic content such as advertising.

Sample A traffic instance generated when a browser displays a webpage.

Label A unique identifier assigned to sets of webpages which are the same. For example, webpages differing only in advertising receive the same label. Samples are labeled according to the webpage in the sample's traffic.

Website A set of webpages such that the URLs which cause each webpage to load in the browser are hosted at the same domain.

Site Graph A graph representing the link structure of a website. Nodes correspond to labels assigned to webpages within the website. Edges correspond to links, represented as the set of labels reachable from a given label.

3.1 Label and Site Graph Generation

Although initially appealing, URLs are poorly suited to labeling webpages within a website. URLs may contain arguments which do not impact content and result in different labels aliasing webpages which are the same. URL redirection further complicates labeling, allowing the same URL to refer to multiple webpages (e.g. error pages or A/B testing). Similar challenges affect web crawlers, creating an infinite web of dynamically generated pages [16]. We present a labeling solution based on URLs and designed to accommodate these challenges.

Our approach contains two phases, each composed of a crawling and a graph building step. The crawling step systematically visits the website to gather URLs and record links. The graph building step uses a *canonicalization function* that transforms webpage URLs into labels and generates a graph representing the structure of the website. The URLs observed and site graph produced in the first phase guide the second, larger crawl which is necessary to observe the breadth of non-deterministic URL redirections. We present our approach below.[4]

Execute Initial Crawl The crawl can be implemented as either a depth- or breadth-first search, beginning at the homepage and exploring every link on a page up to a fixed maximum depth. We perform a breadth first search to depth 5. This crawl will produce a graph $G = (U, E)$, where U represents the set of URLs seen as links during the crawl, and $E = \{(u, u') \in U \times U \mid u \text{ links to } u'\}$ represents links between URLs in U.

Canonicalize Initial Graph First, we construct a canonicalization function of the form $C : U \rightarrow L$, where C denotes the canonicalization function, U denotes the initial set of URLs, and L denotes the set of labels. We then use our canonicalization function to produce an initial site graph $G' = (L, E')$ where L represents the set of labels on the website and E' represents links. We construct E' as follows:

$$E' = \{(C(u), C(u')) \mid (u, u') \in E\} \tag{1}$$

We define a reverse canonicalization function $R : L \rightarrow \mathcal{P}(U)$ such that

$$R(l) = \{u \in U \mid C(u) = l\} \tag{2}$$

Note that $\mathcal{P}(X)$ denotes the *power set* of X, which is the set of all subsets of X.

Execute Primary Crawl The primary crawl allows the attacker to more fully observe the URL redirection behavior of the website. The attacker conducts the primary crawl in a series of browsing sessions; we fixed the length of each session to 75 labels. The attacker builds browsing sessions using a random walk through G', prioritizing labels not yet visited. The attacker then executes each browsing session by visiting a URL u for each label l such that $u \in R(l)$. The attacker records the value of `document.location` once u and all associated resources are done loading to identify any URL redirections. U' denotes the set of final URLs which are observed in `document.location`. We define a new function $T : U \rightarrow \mathcal{P}(U')$ such that $T(u) = \{u' \in U' \mid u \text{ resolved at least once to } u'\}$. We use this to define a new translation $T' : L \rightarrow \mathcal{P}(U')$ such that

$$T'(l) = \bigcup_{u \in R(l)} T(u) \tag{3}$$

[4] A more detailed description of the crawling infrastructure, canonicalization process and graph generation is available on arXiv [17].

	Initial Site Graph G'			Selected Subset			Final Site Graph G'''	
Website	URLs	Labels	Avg. Links	URLs	Labels	Avg. Links	Labels	Avg. Links
ACLU	54398	28383	130.5	1061	500	41.7	506	44.7
Bank of America	1561	613	30.2	1105	500	30.3	472	43.2
Kaiser Permanente	1756	780	29.7	1030	500	22.6	1037	141.1
Legal Zoom	5248	3973	26.8	602	500	11.8	485	12.2
Mayo Clinic	33664	33094	38.1	531	500	12.5	990	31.0
Netflix	8190	5059	13.8	2938	500	6.2	926	9.0
Planned Parenthood	6336	5520	29.9	662	500	24.8	476	24.4
Vanguard	1261	557	28.4	1054	500	26.7	512	30.8
Wells Fargo	4652	3677	31.2	864	500	17.9	495	19.5
YouTube	64348	34985	7.9	953	500	4.3	497	4.24

Table 2: "Selected Subset" denotes the subset of the initial site graph randomly selected for inclusion in our evaluation, "Avg. Links" denotes the average number of links per label, and "URLs" indicates the number of URLs seen as links in the preliminary site graph corresponding to an included label.

Refine Initial Graph To produce the final site graph, we construct a new canonicalization function $C' : U' \rightarrow L'$, where U' denotes the final set of URLs and L' denotes a new set of labels. The final graph G''' must maintain the property that for any browsing path in the initial graph G' and any URL redirections in T', after canonicalization with C' the path must also be valid in the final graph. Therefore, the attacker defines an intermediary graph $G'' = (U', E'')$ such that E'' is defined as

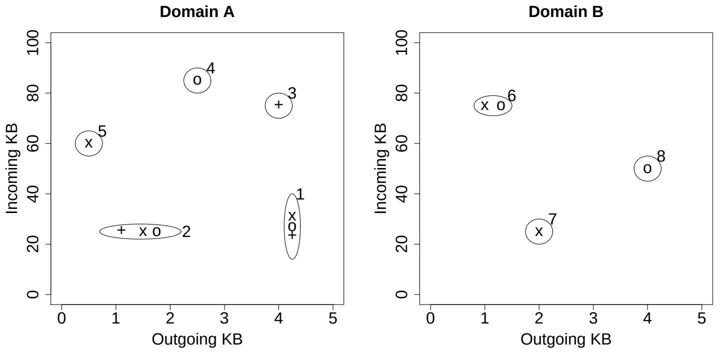
$$E'' = \{(u, u') \mid u \in T'(l) \ \wedge \ u' \in T'(l') \, \forall \, (l, l') \in E'\} \tag{4}$$

We apply our canonicalization function C' to produce a final graph $G''' = (L', E''')$ where

$$E''' = \{(C'(u), C'(u')) \, \forall \, (u, u') \in E''\} \tag{5}$$

guaranteeing that we have strictly increased the connectivity of the site graph.

In the interest of balancing, given available resources, the amount of collection modes, samples of each label, websites surveyed, and pages included from each website, we selected a 500 page subset of each initial site graph. Consequently, we were able to complete data collection in about four weeks during December 2013 and January 2014 using four virtual hosts. We initialize the selected subset to include the label corresponding to the homepage, and iteratively expand the subset by adding a randomly selected label reachable from the selected subset via the link structure of the initial site graph until 500 labels are selected. The set of links for the graph subset is defined as any links occurring between the 500 selected labels. Table 2 presents properties of the initial site graph G', selected subset, and the final site graph G''' for each of the 10 websites we survey. Note that the second crawl also serves as the data collection process; samples are labeled as $C'(u') \in L'$ where u' denotes the value of `document.location` when the sample finished loading. Each model uses only redirections observed in training data when generating the site graph used by the HMM for that model.



(a) (b)

Burst Pairs (KB)

Domain	A	B
Sample x	(0.5, 60) (1.6, 22) (4.2, 30)	(1.1, 75) (2.0, 25)
Sample +	(1.1, 22) (4.0, 75) (4.2, 21)	
Sample o	(1.8, 22) (2.4, 83) (4.2, 25)	(1.4, 75) (4.0, 50)

(c)

Feature Values

Domain	A					B		
Index	1	2	3	4	5	6	7	8
Sample x	.9	1	0	0	1	.8	1	0
Sample +	.9	.8	1	0	0	0	0	0
Sample o	1	.9	0	1	0	.8	0	1

(d)

Fig. 2: Table 2c displays the burst pairs extracted from three hypothetical samples. Figures 2a and 2b show the burst pair clusters. Figure 2d depicts the Bag-of-Gaussians features for each sample. Our Gaussian similarity metric enables our attack to distinguish minor traffic variations from significant differences.

3.2 Feature Extraction and Machine Learning

This section presents our individual sample classification technique. First, we describe the information which we extract from a sample, then we describe processing to produce features for machine learning, and finally describe the application of the learning technique itself.

We initially extract traffic burst pairs from each sample. Burst pairs are defined as the collective size of a contiguous outgoing packet sequence followed by the collective size of a contiguous incoming packet sequence. Intuitively, contiguous outgoing sequences correspond to requests, and contiguous incoming sequences correspond to responses. All packets must occur on the same TCP connection to minimize the effects of interleaving traffic. For example, denoting outgoing packets as positive and incoming packets as negative, the sequence `[+1420, +310, -1420, -810, +530, -1080]` would result in the burst pairs `[1730, 2230]` and `[530, 1080]`. Analyzing traffic bursts removes any fragmentation effects. Additionally, treating bursts as pairs allows the data to contain minimal ordering information and go beyond techniques which focus purely on packet size distributions.

Once burst pairs are extracted from each TCP connection, the pairs are grouped using the second level domain of the host associated with the destination IP of the connection. All IPs for which the reverse DNS lookup fails are treated as a single "unknown" domain. Pairs from all samples from each domain undergo k-means clustering to identify commonly occurring and closely related tuples. We fit a Gaussian distribution to each cluster using a maximum likelihood estimates of the mean and covariance.[5] We then treat each cluster as a feature dimension, producing our fixed width feature vector. Features are extracted from samples by computing the extent to which each Gaussian is represented in the sample.

Figure 2 depicts the feature extraction process using a fabricated example involving three samples and two domains. Clustering results in five clusters for Domain A and three clusters for Domain B, ultimately producing an eight-dimensional feature vector. Sample `x` has traffic tuples in clusters 1, 2, 5, 6 and 7, but no traffic tuples in clusters 3, 4, 8, so its feature vector has non-zero values in dimensions 1, 2, 5, 6, 7, and zero values in dimensions 3, 4, 8. We create feature vectors for samples `+` and `o` in a similar fashion.

Analogously to the Bag-of-Words document processing technique, our approach projects a variable length vector of tuples into a finite dimensional space where each dimension "occurs" to some extent in the original sample. Whereas occurrence is determined by word count in Bag-of-Words, occurrence in our method is determined by Gaussian likelihood. For this reason, we refer to our approach as Bag-of-Gaussians (BoG).

We specify our approach formally as follows:

- Let X denote the entire set of tuples from a sample, with $X^d \subseteq X$ denoting the set all tuples observed at domain d.
- Let Σ_i^d, μ_i^d denote the covariance and mean of Gaussian i at domain d.
- Let F denote all features, with F_i^d denoting feature i from domain d.

$$F_i^d = \sum_{x \in X^d} \mathcal{N}(x | \Sigma_i^d, \mu_i^d) \tag{6}$$

To determine the best number of Gaussian features for each domain, we divide the training data into two parts to train and evaluate models using K values of 4000, 1000 and 500. We then retrain using all training data and the best performing K values for each domain.

Once Gaussian features have been extracted from each sample the feature set is augmented to include counts of packet sizes observed over the entire trace. For example, if the lengths of all outgoing and incoming packets are between 1 and 1500 bytes, we add 3000 additional features where each feature corresponds to the total number of packets sent in a specific direction with a specific size. We linearly normalize all features to be in the range $[0, 1]$ and train a model using L2 regularized multi-class logistic regression with the `liblinear` package [18]. We use $C = 128$ for all sites as we observed varying C did not improve accuracy enough for any site to justify the additional computational cost.

[5] For clusters where all samples occur at the same point, we set the covariance matrix to a scalar matrix with $\lambda = N^{-1}$, where N denotes the size of the cluster.

3.3 Hidden Markov Model

The basic attack presented in section 3.2 classifies each sample independently. In practice, samples in a browsing session are not independent since the link structure of the website guides the browsing sequence. We leverage this ordering information, contained in the site graph produced in section 3.1, to improve results using a hidden Markov model (HMM). Recall that a HMM for a sequence of length N is defined by a set of latent (unknown) variables $Z = \{z_n \mid 1 \leq n \leq N\}$, a set of observed variables $X = \{x_n \mid 1 \leq n \leq N\}$, transition matrix A such that $A_{i,j} = P(z_{n+1} = j | z_n = i)$, an initial distribution π such that $\pi_j = P(z_1 = j)$ and an emission function $E(x_n, z_n) = P(x_n | z_n)$.

To apply the HMM, we treat sample labels as latent variables and the traffic contained in the samples as observed variables. We then use the Viterbi algorithm to find the most likely sequence of labels Z visited by a user given the observed traffic X produced by the user. Given a traffic sample, the logistic regression model specifies the likelihood of each label and acts as the emission function E required by the Viterbi algorithm. We assume in the initial distribution π that the user is equally likely to begin browsing with any label in the website, and construct the transition matrix A such that if the site graph contains a link from label i to label j, then $A_{i,j} = N_i^{-1}$, where N_i denotes the number of links leading from label i. If there is no link leading from label i to label j, then $A_{i,j} = 0$.

4 Impact of Evaluation Conditions

In this section we demonstrate the impact of evaluation conditions on accuracy results and traffic characteristics. First, we present the scope, motivation and experimental methodology of our investigation. Then, we present the results of our experiments on four attack implementations, with the most affected attack decreasing accuracy from 68% to 51%. We discuss attack accuracy only insofar as is necessary to understand the impact of evaluation conditions; we defer a full attack evaluation to section 5.

Cache Configuration The browser cache improves page loading speed by storing previously loaded web resources; this poses two challenges to traffic analysis. Providing content locally decreases the total amount of traffic, reducing the information available for use in an attack. Additionally, differences in browsing history cause differences in cache contents and further vary network traffic. Since privacy tools such as Tor frequently disable caching, many prior evaluations have disabled caching as well [19]. General HTTPS users are unlikely to modify cache settings, so we evaluate the impact of enabling caching to default settings.

User-Specific Cookies If an evaluation collects all data with either the same browser instance or repeatedly uses a fresh browser image, there are respective assumptions that the attacker and victim will either share the same cookies or use none at all. While a traffic analysis attacker will not have access to victim cookies, privacy technologies which begin all browsing sessions from a clean browsing image effectively share the *null cookie*. We compare the performance of evaluations which use the same (non-null) cookie value in all data, different (non-null) cookie values in training and evaluation, a null cookie in all data, and evaluations which mix null and non-null cookies.

Pageview Diversity Many evaluations collect data by repeatedly visiting a fixed set of URLs from a single machine and dividing the collected data for training and evaluation. This approach introduces an unrealistic assumption that an attacker will be able to collect separate training data for each victim, visiting the exact same set of webpages as the victim. We examine the impact of allowing the victim to intersperse browsing of multiple websites, including websites outside our attacker's monitoring focus.[6]

Webpage Similarity Since HTTPS usually allows an eavesdropper to learn the domain a user is visiting, our evaluation focuses on differentiating individual webpages within a website protected by HTTPS. Differentiating webpages within the same website may pose a greater challenge than differentiating website homepages. Webpages within the same website share many resources, increasing caching and decreasing data available for analysis. We examine the relative traffic volumes of browsing both website homepages and webpages within a website.

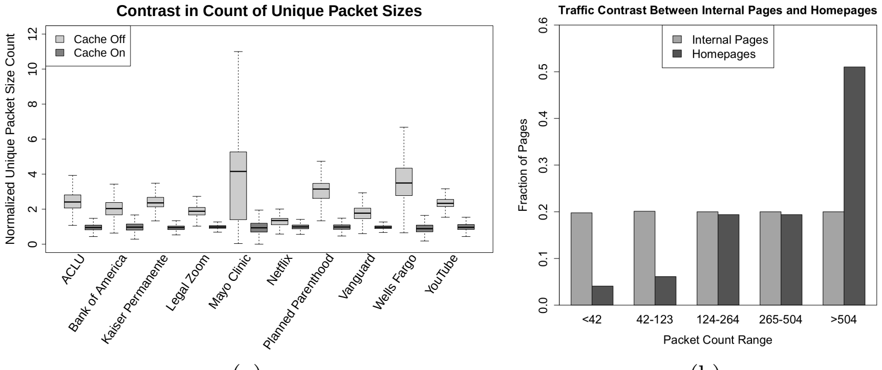
To quantify the impact of evaluation conditions on accuracy results, we design four modes for data collection designed to isolate specific effects. Our approach assumes that data will be gathered in a series of browsing sessions, each consisting of a fixed number of samples. The four modes are as follows:

1. Cache disabled, new virtual machine (VM) for each browsing session
2. Cache enabled, new VM for each browsing session
3. Cache enabled, persistent VM for all browsing sessions, single site per VM
4. Cache enabled, persistent VM for all browsing sessions, all sites on same VM

We fixed the session length to 75 samples and collected at least 16 samples of each label under each collection mode. The first two modes differ only with respect to cache configuration and begin each browsing session with a fresh VM image to eliminate any cookie persistence in browser or machine state. In effect the second, third and fourth modes each represent a distinct cookie value, with the second mode representing a null cookie and the third and fourth modes having actual, distinct, cookie values set by the site. The third and fourth modes differ in pageview diversity. In the context of HTTPS evaluations, the fourth mode most closely reflects the behavior of actual users and hence serves as evaluation data, while the second and third modes generate training data.

Our analysis reveals that caching significantly decreases the number of unique packet sizes observed for samples of a given label. We focus on the number of unique packet sizes since packet size counts are a commonly used feature in traffic analysis attacks. Figure 3a contrasts samples from the first and second collection modes, presenting the effect of caching on the number of unique packet sizes observed per label for each of the 10 websites we evaluate. Note that the

[6] This is different from the open-world vs. closed-world distinction in section 2, as we assume that the attacker will train a model for each website in its entirety and identify the correct model based on traffic destination. Here, we are concerned with effects on browser cache or personalized content which may impact traffic analysis.



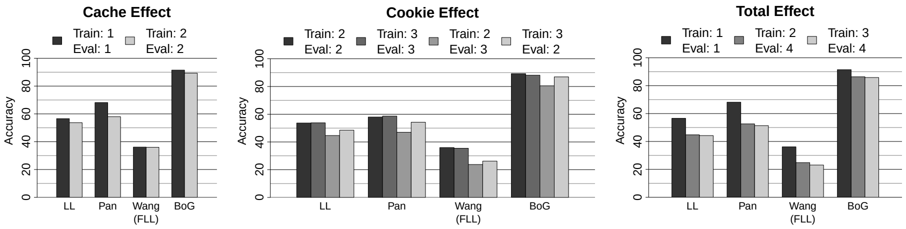
(a) (b)

Fig. 3: Impact of evaluation conditions on traffic characteristics. Figure 3a presents the increase in number of unique packet sizes per sample of a given label caused by disabling the cache. For each label l we determine the mean number l_m of unique packet sizes for samples of l with caching enabled, and normalize the unique packet size counts of all samples of label l using l_m. We present the normalized values for all labels separated by cache configuration. Figure 3b presents the decrease in traffic volume caused by browsing webpages internal to a website as compared to website homepages. Similar to the effect of caching, the decreased traffic volume is likely to increase classification errors.

figure only reflects TCP data packets. We use a normalization process to present average impact of caching on a *per-label basis* across an *entire website*, allowing us to depict for each website the expected change in number of unique packet sizes for any given label as a result of disabling the cache.

Since prior works have focused largely on website homepages, we present data demonstrating a decrease in traffic when browsing webpages within a website. Figure 3b presents the results of browsing through the Alexa top 1,000 websites, loading the homepage of each site, and then loading nine additional links on the site at random with caching enabled. By partitioning the total count of data packets transferred in the loading of webpages internal to a website into five equal size buckets we see that there is a clear skew towards homepages generating more traffic. Similar to the traffic increase from disabled caching, the increased traffic of website homepages is likely to increase accuracy.

Beyond examining traffic characteristics, our analysis shows that factors such as caching, user-specific cookies and pageview diversity impact attack accuracy measurements. We examine each of these factors by training a model using data from a specific collection mode, and comparing model accuracy when evaluated on a range of collection modes. Since some models must be trained and evaluated using the same collection mode we must select a portion of the data from each mode for training and leave the remainder for evaluation. We perform a three-fold evaluation for each attack, varying the evaluation data used for each fold. Figure 4 presents the results of our analysis:



(a) (b) (c)

Mode Number	Cache	Cookie Retention	Browsing Scope
1	Disabled	Fresh VM every 75 samples	Single website
2	Enabled	Fresh VM every 75 samples	Single website
3	Enabled	Same VM for all samples	Single website
4	Enabled	Same VM for all samples	All websites

(d)

Fig. 4: "Train: X, Eval: Y" indicates training data from mode X and evaluation data from mode Y as shown in Table 4d. For evaluations which use a privacy tool such as the Tor browser bundle and assume a closed world, training and evaluating using mode 1 is most realistic. However, in the HTTPS context training using mode 2 or 3 and evaluating using mode 4 is most realistic. Figure 4c presents differences as large as 17% between these conditions, demonstrating the importance of evaluation conditions when measuring attack accuracy.

Cache Effect Figure 4a compares the performance of models trained and evaluated using mode 1 to models trained and evaluated using mode 2. As these modes differ only by enabled caching, we see that caching has moderate impact and can influence reported accuracy by as much as 10%.

Cookie Effect Figure 4b measures the impact of user-specific cookies by comparing the performance of models trained and evaluated using browsing modes 2 and 3. We observe that both the null cookie in mode 2 and the user-specific cookie in mode 3 generally perform 5-10 percentage points better when the evaluation data is drawn from the same mode as the training data. This suggests that any difference in cookies between training and evaluation conditions will impact accuracy results.

Total Effect Figure 4c presents the combined effects of enabled caching, user-specific cookies and increased pageview diversity. Recalling Figure 4b, notice that models trained using mode 2 perform similarly on modes 3 and 4, and models trained using mode 3 perform similarly on modes 2 and 4, confirming the importance of user-specific cookies. In total, the combined effect of enabled caching, user-specific cookies and pageview diversity can influence reported accuracy by as much as 17%. Figure 4b suggests that the effect is primarily due to caching and cookies since mode 2 generally performs better on mode 4, which includes visits to other websites, than on mode 3, which contains traffic from only a single website.

5 Attack Evaluation

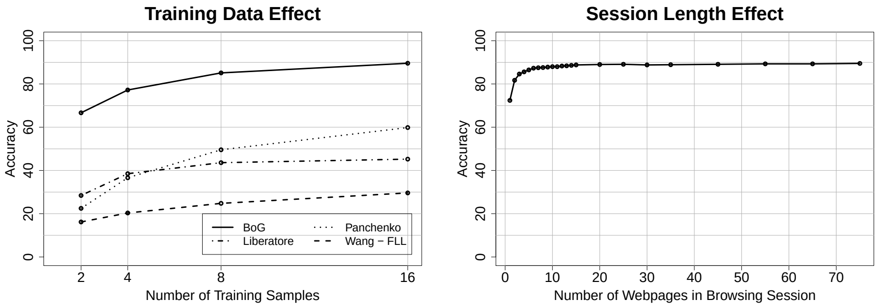
In this section we evaluate the performance of our attack. We begin by presenting the selection of previous techniques for comparison and the implementation of each attack. Then, we present the aggregate performance of each attack across all 10 websites we consider, the impact of training data on attack accuracy, and the performance each attack at each individual website.

We select the Liberatore and Levine (LL), Panchenko *et al.* (Pan), and Wang *et al.* attacks for evaluation in addition to the BoG attack. The LL attack offers a view of baseline performance achievable from low level packet inspection, applying naive Bayes to a feature set consisting of packet size counts [6]. We implemented the LL attack using the naive Bayes implementation in `scikit-learn` [20]. The Pan attack extends size count features to include additional features related to burst length as measured in both packets and bytes as well as total traffic volume [8]. For features aggregated over multiple packets, the Pan attack rounds feature values to predetermined intervals. We implement the Pan attack using the `libsvm` [21] implementation of the RBF kernel support vector machine with the C and γ parameters specified by Panchenko. We select the Pan attack for comparison to demonstrate the significant benefit of Gaussian similarity rather than predetermined rounding thresholds. The BoG attack functions as described in section 3. We implement the BoG attack using the k-means package from `sofia-ml` [22] and logistic regression with class probability output from `liblinear` [18], with `Numpy` [23] performing intermediate computation.

The Wang attack assumes a fundamentally different approach from LL, Pan and BoG based on string edit distance [9]. There are several variants of the Wang attack which trade computational cost for accuracy by varying the string edit distance function. Wang reports that the best distance function for raw packet traces is the Optimal String Alignment Distance (OSAD) originally proposed by Cai *et al.* [4]. Unfortunately, the edit distance must be computed for each pair of samples, and OSAD is extremely expensive. Therefore, we implement the Fast Levenshtein-Like (FLL) distance,[7] an alternate edit distance function proposed by Wang which runs approximately 3000x faster.[8] Since Wang demonstrates that FLL achieves 46% accuracy operating on raw packet traces, as compared to 74%

[7] Note that the original attack rounded packet sizes to multiples of 600; we operate on raw packet sizes as we found this improves attack accuracy in our evaluation.

[8] OSAD has $O(mn)$ runtime where m and n represent the length of the strings, whereas FLL runs in $O(m+n)$. Wang *et al.* report completing an evaluation with 40 samples of 100 labels each in approximately 7 days of CPU time. Since our evaluation involves 10 websites with approx. 500 distinct labels each and 16 samples of each label for training and evaluation, we would require approximately 19 months of CPU time (excluding any computation for sections 4 or 6).



(a)

(b)

Fig. 5: Performance of BoG attack and prior techniques. Figure 5a presents the performance of all four attacks as a function of training data. Figure 5b presents the accuracy of the BoG attack trained with 16 samples as a function of browsing session length. At one sample, the HMM has no effect and reveals the effectiveness of the BoG attack without the HMM. Note that the BoG attack achieves 90% accuracy as compared to 60% accuracy with the best prior work.

accuracy with OSAD, we view FLL as a rough indicator of the potential of the OSAD attack. We implement the Wang - FLL attack using `scikit-learn` [20].

We now examine the performance of each attack implementation. We evaluate attacks using data collected in mode 4 since this mode is most similar to the behavior of actual users. We consider both modes 2 and 3 for training data to avoid any bias introduced by using the same cookies as seen in evaluation data or browsing the exact same websites. As shown in Figure 4, mode 2 performs slightly better so we train all models using data from mode 2.

Consistent with prior work, our evaluation finds that accuracy of each attack improves with increased training data, as indicated by Figure 5a. Note that since we only collect 16 samples of each label in each collection mode, we are unable to conduct a multi-fold evaluation since all data is required for a single 16 training sample model. Notice that the Pan attack is most sensitive to variations in the amount of training data, and the BoG attack continues to perform well even at low levels of training data. In some cases an attacker may have ample opportunity to collect training data, although in other cases the victim website may attempt to actively resist traffic analysis attacks by detecting crawling behavior and engaging in cloaking, rate limiting or other forms of blocking.

The BoG attack derives significant performance gains from the application of the HMM. Figure 5b presents the BoG attack accuracy as a function of the browsing session length. Although we collect browsing sessions which each contain 75 samples, we simulate shorter browsing sessions by applying the HMM to randomly selected subsets of browsing sessions and observing impact on accuracy. At session length 1 the HMM has no effect and the BoG attack achieves 71% accuracy, representing the improvement over the Pan attack resulting from the Gaussian feature extraction. The HMM accounts for the remaining perfor-

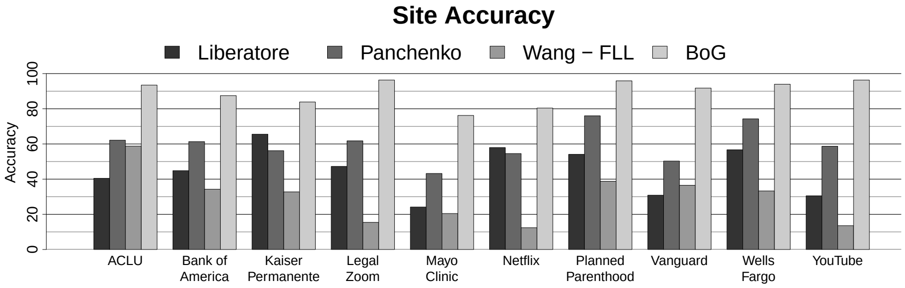
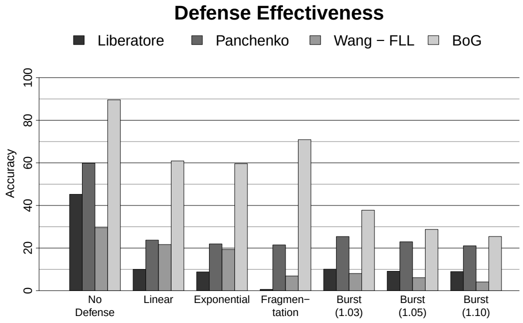


Fig. 6: Accuracy of each attack for each website. Note that the BoG attack performs the worst at Kaiser Permanente, Mayo Clinic and Netflix, which each have approx. 1000 labels in their final site graphs according to Table 2. The increase in graph size during finalization suggests potential for improved performance through better canonicalization to eliminate labels aliasing the same webpages.

mance improvement from 71% accuracy to 90% accuracy. We achieve most of the benefit of the HMM after observing two samples in succession, and the full benefit after observing approximately 15 samples. Although any technique which assigns a likelihood to each label for each sample could be extended with a HMM, applying a HMM requires solving the labeling and site graph challenges which we present novel solutions for in section 3.

Although the BoG attack averages 90% accuracy overall, only 4 of the 10 websites included in evaluation have accuracy below 91%. Figure 6 presents the accuracy of each attack at each website. The BoG attack performs the worst at Mayo Clinic, Netflix and Kaiser Permanente. Notably, the number of labels in the site graphs corresponding to each of these websites approximately doubles during the finalization process summarized in Table 2 of section 3. URL redirection causes the increase in labels, as new URLs appear whose corresponding labels were not included in the preliminary site graph. Some new URLs may have been poorly handled by the canonicalization function, resulting in labels which alias the same content. Although we collected supplemental data to gather sufficient training samples for each label, the increase in number of labels and label aliasing behavior degrade measured accuracy for all attacks.

Despite the success of string edit distance based attacks against Tor, the Wang - FLL attack struggles in the HTTPS setting. Wang's evaluation is confined to Tor, which pads all packets into fixed size cells, and does not effectively explore edit distance approaches applied to unpadded traffic. Consistent with the unpadded nature of HTTPS, we observe that Wang's attack performs best on unpadded traffic in the HTTPS setting. Despite this improvement, the Wang - FLL technique may struggle because edit distance treats all unique packet sizes as equally dissimilar; for example, 310 byte packets are equally similar to 320 byte packets and 970 byte packets. Additionally, the application of edit distance at the packet level causes large objects sent in multiple packets to have proportionally large impact on edit distance. This bias may be more apparent in the HTTPS context than with website homepages since webpages within the same



(a)

Defense	Byte Overhead	Packet Overhead	Burst Entropy
No Defense	1.000	1.000	8.164
Linear	1.032	1.000	5.309
Exponential	1.055	1.000	5.241
Fragmentation	1.000	1.450	8.164
Burst: 1.03	1.025	1.022	5.994
Burst: 1.05	1.043	1.032	5.568
Burst: 1.10	1.090	1.065	5.010

(b)

Fig. 7: Figure 7a presents the impact of defenses on attack accuracy. Figure 7b presents defense costs and entropy of burst sizes. The Burst defense is a novel proposal, substantially decreasing accuracy at a cost comparable to a low level defense. Entropy provides useful but limited insight into defense effectiveness, as rare values minimally impact entropy but may uniquely identify content.

website are more similar than homepages of different websites. Replacing the FLL distance metric with OSAD or Damerau-Levenshtein would improve attack accuracy, although the poor performance of FLL suggests the improvement would not justify the cost given the alternative techniques available.

6 Defense

This section presents and evaluates several defense techniques, including our novel Burst defense which operates between the application and TCP layers to obscure high level features of traffic while minimizing overhead. Figure 7 presents the impacts and costs of the defenses we consider. We find that evaluation context significantly impacts defense performance, as we observe increased effectiveness of low level defenses in our evaluation as compared to prior work [5]. Additionally, we find that the Burst defense offers significant performance improvements while maintaining many advantages of low level defense techniques.

We select defenses for evaluation on the combined basis of cost, deployability and effectiveness. We select the linear and exponential padding defenses from Dyer *et al.* as they are reasonably effective, have the lowest overhead, and are implemented statelessly below the TCP layer. The linear defense pads all packet sizes up to multiples of 128, and the exponential defense pads all packet sizes up to powers of 2. Stateless implementation at the IP layer allows for easy adoption across a wide range of client and server software stacks. Additionally, network overhead is limited to minor increases in packet size with no new packets generated, keeping costs low in the network and on end hosts. We also introduce the fragmentation defense which randomly splits any packet which is smaller than

Algorithm 1 Threshold Calculation

```
Precondition: $bursts$ is a set containing the length of each burst in a given direction
    in defense training traffic
Precondition: $threshold$ specifies the maximum allowable cost of the Burst defense
$thresholds \leftarrow set()$
$bucket \leftarrow set()$
for $b$ in $\texttt{sorted}(bursts)$ do
    $inflation \leftarrow \texttt{len}(bucket + b) * \texttt{max}(bucket + b)/\texttt{sum}(bucket + b)$
    if $inflation \geq threshold$ then
        $thresholds \leftarrow thresholds + max(bucket)$
        $bucket \leftarrow set() + b$
    else
        $bucket \leftarrow bucket + b$
    end if
end for
return $thresholds + \texttt{max}(bucket)$
```

Algorithm 2 Burst Padding

```
Precondition: $burst$ specifies the size of a directed traffic burst
Precondition: $thresholds$ specifies the thresholds obtained in Algorithm 1
for $t$ in $\texttt{sorted}(thresholds)$ do
    if $t \geq burst$ then
        return $t$
    end if
end for
return $burst$
```

the MTU, similar to portions of the strategy adopted by HTTPOS [14]. Fragmentation offers the deployment advantages of not introducing any additional data overhead, as well as being entirely supported by current network protocols and hardware. We do not consider defenses such as BuFLO or HTTPOS given their complexity, cost and the effectiveness of the alternatives we do consider [5,14].

The exponential defense slightly outperforms the linear defense, decreasing the accuracy of the Pan attack from 60% to 22% and the BoG attack from 90% to 60%. Notice that the exponential defense is much more effective in our evaluation context than Dyer's context, which focuses on comparing website homepages loaded over an SSH tunnel with caching disabled and evaluation traffic collected on the same machine as training traffic. The fragmentation defense is extremely effective against the LL and Wang - FLL attacks, reducing accuracy to below 1% and 7% for each respective attack, but less effective against the Pan and BoG attacks as these attacks perform TCP stream reassembly. Since TCP stream reassembly is expensive and requires complete access to traffic, the fragmentation defense may be a superior choice against many adversaries in practice.

Although the fragmentation, linear and exponential defenses offer the deployment advantages of functioning statelessly below the TCP layer, their effectiveness is limited. The Burst defense offers greater protection, operating between the TCP layer and application layer to pad contiguous bursts of traffic up to predefined thresholds uniquely determined for each website. Reducing the number of thresholds allows the Burst defense to achieve greater privacy at the expense of increased padding.

Algorithms 1 and 2 present the training and application of the Burst defense respectively. Unlike the BoG attack which considers bursts as a tuple, for the purposes of the Burst defense (and Figure 7b) we define a burst as a contiguous sequence of packets in the same direction on the same TCP connection. Hence, we apply Algorithm 1 in each direction. We evaluate the Burst defense for *threshold* values 1.03, 1.05 and 1.10, with the resulting cost and performance shown in Figure 7. The Burst defense outperforms defenses which operate solely at the packet level by obscuring features aggregated over entire TCP streams. Simultaneously, the Burst defense offers deployability advantages over techniques such as HTTPOS since the Burst defense is implemented between the TCP and application layers. The cost of the Burst defense compares favorably to defenses such as HTTPOS, BuFLO and traffic morphing, reported to cost at least 37%, 94% and 50% respectively [4,15].

7 Discussion and Conclusion

This work examines the vulnerability of HTTPS to traffic analysis attacks, focusing on evaluation methodology, attack and defense. Although we present novel contributions in each of these areas, many open problems remain.

Our examination of evaluation methodology focuses on caching and user-specific cookies, but does not explore factors such as browser differences, operating system differences, mobile/tablet devices or network location. Each of these factors may contribute to traffic diversity in practice, likely degrading attack accuracy. Additional future work remains in the area of attack development. To date, all approaches have assumed that the victim browses the web in a single tab and that successive page loads can be easily delineated. Future work should investigate actual user practice in these areas and impact on analysis results. For example, while many users have multiple tabs open at the same time, it is unclear how much traffic a tab generates once a page is done loading. Additionally, we do not know how easily traffic from separate page loadings may be delineated given a contiguous stream of user traffic.

Defense development and evaluation also require further exploration. Attack evaluation conditions and defense development are somewhat related since conditions which favor attack performance will simultaneously decrease defense effectiveness. Defense evaluation under conditions which favor attack creates the appearance that defenses must be complex and expensive, effectively discouraging defense deployment. To increase likelihood of deployment, future work must investigate necessary defense measures under increasingly realistic conditions since realistic conditions may substantially contribute to effective defense.

References

1. A. Hintz, “Fingerprinting Websites Using Traffic Analysis,” in *Proc. Privacy Enhancing Technologies Conference*, 2003.
2. Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, “Statistical Identification of Encrypted Web Browsing Traffic,” in *Proc. IEEE S&P*, 2002.
3. D. Herrmann, R. Wendolsky, and H. Federrath, “Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naive-Bayes Classifier,” in *Proc. of ACM CCSW*, 2009.
4. X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, “Touching From a Distance: Website Fingerprinting Attacks and Defenses,” in *Proc. of ACM CCS*, 2012.
5. K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail,” in *IEEE S&P*, 2012.
6. M. Liberatore and B. N. Levine, “Inferring the Source of Encrypted HTTP Connections,” in *Proc. ACM CCS*, 2006.
7. G. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, “Privacy Vulnerabilities in Encrypted HTTP Streams,” in *Proc. PETs Workshop*, May 2005.
8. A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, “Website Fingerprinting in Onion Routing Based Anonymization Networks,” in *Proc. ACM WPES*, 2011.
9. T. Wang and I. Goldberg, “Improved Website Fingerprinting on Tor,” in *Proc. of ACM WPES 2013*.
10. H. Cheng and R. Avnur, “Traffic Analysis of SSL Encrypted Web Browsing,” http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps, 1998.
11. G. Danezis, “Traffic Analysis of the HTTP Protocol over TLS,” http://research.microsoft.com/en-us/um/people/gdane/papers/TLSanon.pdf.
12. S. Chen, R. Wang, X. Wang, and K. Zhang, “Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow,” in *Proc. IEEE S&P*, 2010.
13. S. E. Coull, M. P. Collins, C. V. Wright, F. Monrose, and M. K. Reiter, “On Web Browsing Privacy in Anonymized NetFlows,” in *Proc. USENIX Security*, 2007.
14. X. Luo, P. Zhou, E. W. W. Chan, W. Lee, R. K. C. Chang, and R. Perdisci, “HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows.” in *Proc. of NDSS*, 2011.
15. C. V. Wright, S. E. Coull, and F. Monrose, “Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis,” in *NDSS*, 2009.
16. “To infinity and beyond? No!” http://googlewebmastercentral.blogspot.com/2008/08/to-infinity-and-beyond-no.html.
17. B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar, “I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis,” http://arxiv.org/abs/1403.0297, 2014.
18. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A Library for Large Linear Classification,” *JMLR*, no. 9, pp. 1871–1874, 2008.
19. “Torbutton FAQ,” https://www.torproject.org/torbutton/torbutton-options.html.en, Accessed May 2012.
20. scikit-learn. http://scikit-learn.org/stable/.
21. C.-C. Chang and C.-J. Lin, “LIBSVM: A Library for Support Vector Machines,” *ACM Transactions on TIST*, vol. 2, no. 3, 2011.
22. sofia-ml. http://code.google.com/p/sofia-ml/.
23. Numpy. http://www.numpy.org/.